PRIS at 2012 TREC Medical Track:

Query Expansion, Retrieval and Ranking

Jiayue Zhang, Lin Lin, Shudang Diao, Yukun Li, Runnan Liu, Weiran Xu, Jun Guo
School of Information and Communication Engineering
Beijing University of Posts and Telecommunications

1 Data Preprocessing

1.1 XML parsing

The official datasets are XML format so we have to parse them before indexing. We choose Lucene as our tool for indexing and searching ,we select the Jakarta-commons-Digester (the following we referred to as digester) to parse the xml documents.

The xml document is processed by the Digester to be a java object and then we can get the fields that we would use from the java object .In addition, we also process the tag "report_text" in the xml documents so that we can get the age and sexuality information which are very important fields for searching task.

1.2 Negation Detection

People always find some phrases like "did not have head pain" or "there is no pain in your leg"in the medical diagnosis reports .These phrases will make some boring troubles in the medical text retrieval. For example, when we want to find someone who have a headache we may get a report like this

This patient is a**AGE[in 50s]-year-old male with a past medical history of multiple transplants including small bowel, liver, and pancreas in 1998 and status post kidney transplant in 2006, presents with fever. The patient states he woke this morning and thought to have fevers and chills. He also has had some vomiting and diarrhea. Denies any belly pain. He states he feels a little short of breath. He denies any chest pain. No sore throat. No headache.....

In fact, this patient just has fevers and chills. To solve this problem, we use the famous NegEx algorithm .NegEx [5] algorithm is mostly known to Text Mining researchers for finding terms used in negative senses. While, there is a java class to implement Wendy Chapman's NegEx algorithm. This class' author is Junebae Kye .On the base of this class, we write a program to finish the negation detection work and the result show us that this method takes us a better performance.

2 Indexing

Model main component is a search engine based on Apache Lucene. Lucene is a powerful Java library that lets you easily add document retrieval to any application. In recent years Lucene has become exceptionally popular and is now the most widely used information retrieval library

We utilized Lucene for indexing purpose. Lucene provided the function to achieve this goal. Documents and fields are Lucene's fundamental units of indexing and searching. A document is Lucene's atomic unit of indexing and searching. It is a container that holds one or more fields, which in turn contain the real content. Each field has a name to identify it, a text or binary value, and a series of detailed options that describe what Lucene should do with the field value. We use the "age","sex","icd9 code"...as the fields to build the index. This process is not very difficult.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **NOV 2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE **PRIS at 2012 TREC Medical Track: Query Expansion, Retrieval and Ranking**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Beijing University of Posts and Telecommunications,School of Information and Communication Engineering,Beijing, P.R. China, 100876,**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the Twenty-First Text REtrieval Conference (TREC 2012) held in Gaithersburg, Maryland, November 6-9, 2012. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA). U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**3**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

2. Concept-based Query Expansion

In medical search, we have the challenge of ‘semantic gap’, that is, the vocabulary mismatch between relevant documents and topic description. A large part of the reason is the diversification of expression in the medical field. For instance, “Hypertension” in a report has the same meaning with “high blood pressure” in the topic plain text. The presence of a certain organism in a report may also denote a certain disease described in a topic. So it’s import for us to expand the query with expressions that refer to the same meaning.

We expanded our queries with the help of UMLS (Unified Medical Language System) meta-thesaurus and SNOMED medical domain knowledge. First, we used the Meta-Map program to extract UMLS Meta-thesaurus concepts associated with the original query. Second, we mapped the concepts to their SNOMED-CT equivalents using the UMLS Meta-thesaurus. Then, we had our query concepts expanded with SNOMED-CT descriptions. Now, each query concept is replaced by a group of thesauruses. We call it concept group.

3. Queries Construction

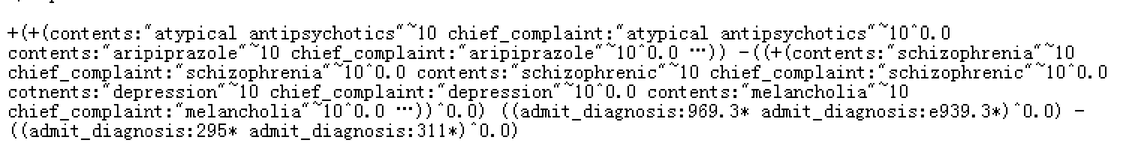
Queries were constructed for Lucene to search different texts against various indexed fields. Each query is consisted of a collection of filters and clauses containing subqueries. （Each subquery contained search terms for only one field, and most of the subqueries had a boost applied to them in order to improve precision by keeping certain clauses from dominating the scoring algorithm.） In different runs, we had different ways to generate the boosts. Table 1 shows the fields and their relations to the parent-query.

Table 1. Query fields and relations

Fields	Relations
contents	MUST/MUST_NOT
chief_complaint	MUST/MUST_NOT
admit_diagnosis	SHOULD/MUST_NOT
age	MUST
sex	MUST

We use negEx to detect the negative contents in the topics. For negative contents, we make the relation of the relate subqueries MUST_NOT.

And each subquery referring to field of contents and chief_complaint consisted of several concept group subqueries connecting with relation MUST. Each concept group subquery was made up of several phrases included in the group with relation SHOULD. Graph 1 shows the construct of one example query.



```
<top>
<num> Number: 179
<desc>
Patients taking atypical antipsychotics without a diagnosis schizophrenia or bipolar depression
</top>

+(+(contents:“atypical antipsychotics”~10 chief_complaint:“atypical antipsychotics”~10^0.0
contents:“aripiprazole”~10 chief_complaint:“aripiprazole”~10^0.0 …)) -((+(contents:“schizophrenia”~10
chief_complaint:“schizophrenia”~10^0.0 contents:“schizophrenic”~10 chief_complaint:“schizophrenic”~10^0.0
cotnents:“depression”~10 chief_complaint:“depression”~10^0.0 contents:“melancholia”~10
chief_complaint:“melancholia”~10^0.0 …))^0.0) ((admit_diagnosis:969.3* admit_diagnosis:e939.3*)^0.0) -
((admit_diagnosis:295* admit_diagnosis:311*)^0.0)
```

Graph 1. Query Construct Example

4. Retrieval

We use Lucene to realize our retrieval. And except for the basic run, our retrieval contained two stages. On the first stage, we retrieve relative reports using the queries generated on the previous step. Then we map the results to visit ids. We set a threshold of 15. When the number of result visit ids was less than the threshold, and no less than one subquery got visit id number which less than 100, we abandoned the subquery which got least visit ids to relax the requirements. Then we go into the second stage of retrieving. We looped this process until we got visit id number greater than the set threshold, or when we had only one subquery left.

5. Learning to rank

In our buptpris_Lrank run, we tried to make the ranking of our results more meaningful. We refer to a semi-supervised approach to learing to rank that uses Boosing[1]. We have 5 features for learning: the Lucene average scores for fields of contents, chief_complaint, admit_diagnosis, sex and age. By this way, we got more suitable boosts for queries.
Because of the limit of deadline, we did't have much research on this method. But it did improve our ranking.

6. Our Runs

The descriptions of our runs:
buptpris_Base: A baseline using Lucene,with query expanded by several tools including MetaMap,UMLS Metathesaurus and SNOMED,and ICD9 information mining.The weight of each indexed field is defined by personal experiences.Result scores computed with lucene retrival scores.
buptpris_Int: To make improvement to buptpris_Base,this run split a query into several subquerys,and compute the intersection of their retrival results.At the same time,this run include a algorithm to deal with the topics returning few results.
buptpris_Cscore: The buptpris_Cscore run considers "contents" field exclusively when getting the final score of each returned visit.With intersection algorithm and few-result-deal algorithm the same as buptprisInt.
buptpris_Lrank: A try to improve the ranking with learning to rank algorithm on the basis of buptpris_Int run.

References

[1]Rong Jin, Hamed Valizadegan, Hang Li, *Ranking Refinement and Its Application for Information Retrieval*, in International Conference on World Wide Web (WWW), 2008.